**信息与软件工程学院**

**基于Python的图像分类项目开发**

**实践总结报告PBLF2021**

**小组成员： 姓名：陈霆润 学号：2020091202003**

**姓名：幸锴文 学号：2020091201022**

**姓名：赵之航 学号：2020091202028**

1. **人工智能概述**
   1. **人工智能的发展与现状**

自2016年谷歌的人工智能AlphaGo在围棋比赛中先后打败李世石和柯洁，人工自智能成为了社会的热点。人工智能的技术成为了科技的热点，人工智能公司成为.了资本的追逐对象。各个行业也开始发展人工智能，而后，熟悉的手机语音助手,人脸识别，购买推荐，自动驾驶都纷纷应用了人工智能。业界将人工智能分为三个阶段，分别是弱人工智能、强人工智能、超人工智能。弱人工智能只能在某一方面超越人类，只能替代某一方面的工作；强人工智能拥有和人类一样的智商，能够在日常生活中代替人类的绝大部分工作；而超人工智能能够像人类一样通过各种采集器、网络进行学习，每天进行多次升级迭代。而在超人工智能阶段，人工智能的水平将远超人类。当今我们的人工智能发展处在弱人工智能阶段，尽管AlphaGo在围棋比赛上打败了世界冠军，但也仅仅限于围棋一方面，并不能在多方面取代人类，甚至在其他领域与人类相差甚远。而现在科学家和各大企业想要实现的目标便是使人工智能达到强人工智能阶段。超人工智能更加接近我们幻想中的神和上帝，离我们现阶段的发展还有较远的距离。

* 1. **人工智能的应用**

现阶段人工智能的几个主要应用有如下几个方面：

* + 1. 无人驾驶汽车

无人驾驶汽车是智能汽车的一种，主要依靠车内以计算机系统为主的智能驾驶控制器来实现无人驾驶。而人工智能主要用于无人驾驶汽车对外界环境的感知、深度学习、共享信息三个方面。

* + 1. 人脸识别

人脸识别是基于人的面部特征进行身份识别的一种生物识别技术。而其中的算机技术和图像处理技术都离不开人工智能。现阶段，人脸识别已经应用于司法、安保、医疗、教育等多个行业。

* + 1. 个性化推荐

个性化推荐是一种基于聚类与协同过滤技术的人工智能应用，它的基础便是海量的数据挖掘，通过对用户多次的输入分析来建立数据模型。在用户下次搜索的过程中为用户的兴趣提供相似的推荐。如淘宝、京东上的相似推荐，每日的新闻推荐等。

* 1. **人工智能技术与分类**

从研究方向上来看，人工智能技术领域包括深度学习技术、、自然语言处理技术、计算机视角技术、智能机器人技术等。

深度学习基于已有的数据进行学习操作，它模拟人脑的机制来解释数据、声音、文本。

自然语言处理技术是用计算机技术进行通讯的一种技术，研究用电子计算机来模拟人的语言的交流过程、目的是使计算机能够了解人类的各种语言，并能够成功的和人类对话。

计算机视角技术是用电脑代替人脑，而用摄像头来代替人眼来对目标进行测量和识别各种图像，并对图像做进一步的处理，使之成为更适合人眼所能识别的图像。

智能机器人技术具备各种各样基于人工智能的传感器和效应器，并配备有“大脑芯片”使其拥有更强的智能性。

1. **图像分类任务分析**
2. **图像分类应用与技术概要**

图像分类项目开发应用分为了三个部分，分别是：

* + 1. 使用预训练模型图像分类
    2. 使用Finetune训练模型
    3. 数据集增强

在第一部分，使用预训练模型图像分类中，我们通过百度的paddlepaddle来调用CNN预训练模型，分别调用了resnet，alexnet，vgg和googlenet四个模型，将图片上传后，对返回的结果进行处理，并对比有效的小狗名称，得出相关的统计量。通过程序运行时间、正确分类的小狗所占百分比、正确分类的非小狗图像所占百分比、正确分类的小狗品种所占百分比以及标签匹配书所占百分比这5个结果进行横向对比。调用seaborn等库来绘制统计图并保存统计图图片。

在第二部分，使用finetune训练模型，通过paddlehub调用googlenet预训练模型，对数据集进行训练。数据集包含五种种类的中药材，分别为百合、党参、枸杞、槐花、金银花。训练后对我们寻找的新的50张图片进行训练，成功识别后将50张图片复制到对应的子文件夹下。

在第三部分，数据集增强板块，引入了numpy、PIL、matplotlib以及math库。以矩阵的方式存储一张图像的每个像素点。通过对矩阵操作，实现了旋转（10°，90°，180°，270°）、平移、镜像（垂直、水平）、放大操作，并将进行操作后的图片保存。对第二部分的数据集中每张图片进行这些操作后，便可以将原数据集进行扩充。将得到的新的数据集再次训练，识别新的50张自己寻找的中药材图片。

1. **图像分类技术**

CNN模型调用：通过paddlepaddle调用CNN模型，有四种：resnet，alexnet，vgg和googlenet。最后通过绘制出的多个条形统计图，综合比较，选出googlenet的识别效果最好和识别效率最高

调用的第三方库：

* 1. math：用于矩阵中角度的计算，将输入的度数通过三角函数转化为需要的值
  2. seaborn：利用狗狗识别的相关结果绘制条形统计图
  3. pillow：生成Image类的实例，用于绘制图像
  4. numpy：用于矩阵的计算，包括利用列表生成矩阵，进行矩阵的乘法
  5. mathplotlib：保存图片，绘制图像
  6. paddlehub：调用CNN模型，调用finetune\_and\_eval接口，创建自定义数据集，生成图像的reader。

调用标准库：

1. os：获取图片的路径，创建路径
2. random：打乱所有图片的路径，随机排序，使数据具有可适性
3. shutil：复制文件

Finetune：调用finetune\_and\_eval接口进行模型的训练

1. **图像分类实现环境**

操作系统：ai studio提供的jupyter notebook

编程环境：

* + 1. 基础版：（用于第一模块）
       1. CPU：2 Cores
       2. RAM：8GB
       3. Disk：100GB
    2. 高级版：（用于finetune）
       1. GPU：Tesla V100. Video
       2. Mem：100GB
       3. CPU：4 Cores
       4. RAM：32GB
       5. Disk：100GB

编程语言：Python

1. **项目实现分析**
2. **任务架构**

第一部分 使用预训练模型图像分类（如图3-1）



图3- 1

第二部分 使用finetune训练模型（如图3-2）

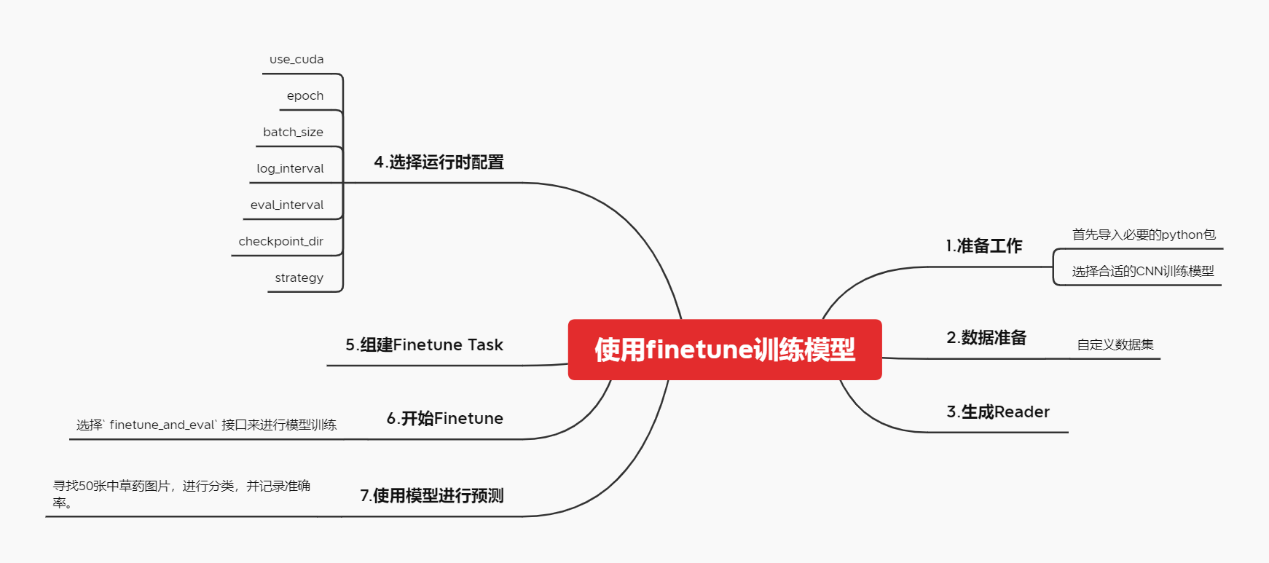


图3- 2

第三部分 数据集增强（如图3-3）

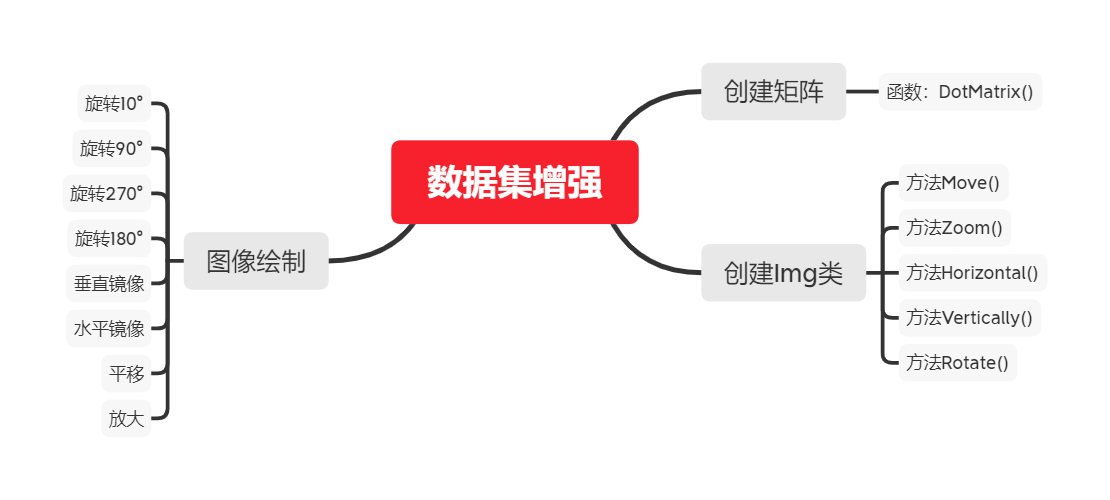


图3- 3

1. **任务模块具体分析及实现**

**3.2.1 使用预训练模型图像分类模块**

TODO1模块

# **TODO:** 1  
start = time()  
end = time()  
total = end - start  
seconds = int((total % 3600)%60)  
minutes = int((total % 3600) / 60)  
hours = int(total / 3600)  
print(f"程序运行时间：{hours}:{minutes}:{seconds}")

在程序刚开始运行时，使用start=time()记录开始时间，在程序运行结束后，使用end=time()记录结束时间，两者之差即为程序运行的时间，再将运行的时间转换为时分秒的形式

程序运行截图：（如图3-4）



图3- 4

TODO2模块

**def** **get\_pet\_labels**():  
 filename\_list = listdir("pet\_image/")  
 petlabels\_dic = {}  
 **for** filename **in** filename\_list:  
 lower\_filename = filename.lower()  
 word\_list\_filename = lower\_filename.split('\_')  
 pet\_name = ''  
 **for** index **in** word\_list\_filename:  
 **if** index.isalpha():  
 pet\_name += (index+" ")  
 pet\_name = pet\_name.strip()  
 petlabels\_dic[filename] = pet\_name  
 **return** petlabels\_dic

调用listdir函数，获得pet\_image文件夹中的所有图像的名称，并以列表的格式返回。

在filename\_list的for循环中，对于每一个filename，即图像的名称（形如：fox\_squirrel\_01.jpg），首先使用函数split将分隔符“\_”分隔，返回一个列表。

使用lower()方法，将字符串全部变为小写。

使用alpha()方法判断字符串是否只有字母而不是数字。

将全是字母的字符串用“+”和空格连接起来，得到最后需要的字符串

将最后得到的字符串作为值，最初的图像名称作为键，放入petlabels\_dic字典中存储，最后返回该数组

TODO3模块

image\_dir = "/home/aistudio/pet\_image/"  
models ={'resnet':'resnet\_v2\_50\_imagenet', 'alexnet': 'alexnet\_imagenet', 'vgg':'vgg19\_imagenet','googlenet':'googlenet\_imagenet','mobilenet':"mobilenet\_v2\_imagenet"}  
**import** paddlehub **as** hub  
  
**def** **classifier**(image\_dir,image\_path,model):  
 module = hub.Module(name=models[model])  
 test\_img\_path =image\_dir+image\_path  
 input\_dict = {"image": [test\_img\_path]}  
 result = module.classification(data=input\_dict)  
 temp\_dict = result[0][0]  
 temp\_list = list(temp\_dict.keys())  
 **return** temp\_list[0]  
**def** **classify\_images**(answers\_dic,model):  
 results\_dic={}  
 petlabel\_dic = {}  
 **for** key,value **in** answers\_dic.items():  
 label = classifier(image\_dir,key,model).lower()  
 petlabel\_dic[key]=label.strip()  
 if\_same = 0  
 found\_idx = petlabel\_dic[key].find(value)  
 **if** found\_idx ==0 **and** len(value)==len(petlabel\_dic[key]):  
 if\_same = 1  
 **if** ((found\_idx == 0) **or** (petlabel\_dic[key][found\_idx-1]==" ")) **and** ((found\_idx+len(value)==len(petlabel\_dic[key]))**or** (petlabel\_dic[key][found\_idx+len(value):found\_idx+len(value)+1]**in** (' ',','))):  
 if\_same = 1   
 found\_idx\_2 = value.find(petlabel\_dic[key])  
 **if** found\_idx\_2 == 0 **and** len(value)==len(petlabel\_dic[key]):  
 if\_same = 1  
 **if** ((found\_idx\_2 == 0) **or** (value[found\_idx\_2-1]==" ")) **and** ((found\_idx\_2 + len(petlabel\_dic[key])== len(value)) **or**  
(value[found\_idx\_2+len(petlabel\_dic[key]):found\_idx\_2+len(petlabel\_dic[key])+1]**in** (' ',','))):  
 if\_same = 1   
 index\_of\_results\_dic = []  
 index\_of\_results\_dic.append(value)  
 index\_of\_results\_dic.append(label)  
 index\_of\_results\_dic.append(if\_same)  
 results\_dic[key] = index\_of\_results\_dic  
 **return** results\_dic

TODO3分为两个函数：classifier函数和classify\_images函数

在classifier函数外部，首先定义好模型的名称，在此处我们使用了5个模型（在后面的结果中发现alexnet准确率极低）。在classfier函数中，依照模型的使用方法，将图片路径放入字典input\_dict中，利用模型识别网络，返回一个结果result。其中，result[0][0]是一个键为图像识别返回的标签的字符串的字典，且字典中只有一个元素。temp\_list = list(temp\_dict.keys())，将键值全部提取出来并转化为列表形式，列表（只有一个元素）中的元素即为图像识别的标签。最后返回这个值。

在classify\_images函数中，考虑到classifier函数一次只能返回一个标签值，所以将classifier函数运用在for循环中。首先创建两个字典，一个为results\_dic，键是宠物图像文件名，值是关于宠物图像的列表，索引0为宠物图像标签，索引1为分类器标签，索引2为标签比较结果，最后作为返回值返回，另一个是petlabel\_dic，键为宠物文件名，值为宠物标签。classify\_images函数的形参有两个，一个是模型名model，另一个是TODO2模块中的answers\_dic。在for循环中，首先传入图像文件夹地址，图像名称，模型名，返回的值进行小写操作，去掉两侧空格操作得到图片识别标签，将图像名作为键，图片识别标签作为值。

考虑到指导PDF中的字符串不匹配的情况有两种，一种原标签为cat，分类器标签为skunk，polecat， wood pussy and lynx catamount，并不匹配。还有一种是原标签为german shepherd dog，分类器标签为german shepherd，应该匹配，但是用指导PDF中的方法并不匹配，所以，不仅在原标签中使用find函数寻找分类器标签，还在分类器标签中使用find函数寻找原标签，满足任意一种即可。最后将宠物图像标签，分类器标签，标签比较结果放入列表index\_of\_results\_dic，再将index\_of\_results\_dic作为值，宠物文件名作为键，放入results\_dic字典中，最后将其返回。

TODO4模块：

#TODO 4  
**def** **adjust\_results4\_isadog**(result\_dic,dogsfile):  
 dognames\_dict = {}  
 **with** open(file=dogsfile) **as** file\_object:  
 **for** line **in** file\_object:  
 dognames\_dict[line.strip()]=1  
 # print(dognames\_dict)  
 **for** key,value **in** result\_dic.items():  
 index\_3 = index\_4 = 0  
 # value[0]是图片的label，value[1]是分类器给出的标签,key\_2是dognames字典的键名  
 **for** key\_2 **in** dognames\_dict.keys():  
 # 首先用value[0]和key\_2比对，然后是value[1]和key\_2比对  
 found\_idx\_1 = key\_2.find(value[0])  
 **if** found\_idx\_1 ==0 **and** len(value[0])==len(key\_2):  
 index\_3 = 1  
 **if** ((found\_idx\_1 == 0) **or** (key\_2[found\_idx\_1-1]==" ")) **and**((found\_idx\_1+len(value[0])==len(key\_2))**or** (key\_2[found\_idx\_1+len(value[0]):found\_idx\_1+len(value[0])+1]**in** (' ',','))):  
 index\_3 = 1   
 found\_idx\_2 = value[0].find(key\_2)  
 **if** found\_idx\_2 ==0 **and** len(value[0])==len(key\_2):  
 index\_3 = 1  
 **if** ((found\_idx\_2 == 0) **or** (value[0][found\_idx\_2-1]==" ")) **and** ((found\_idx\_2 + len(key\_2)== len(value[0])) **or**  
 (value[0][found\_idx\_2+len(key\_2):found\_idx\_2+len(key\_2)+1]**in** (' ',','))):  
 index\_3 = 1   
 # 现在是value[1]和key\_2比对  
 found\_idx\_3 = key\_2.find(value[1])  
 **if** found\_idx\_3 ==0 **and** len(value[1])==len(key\_2):  
 index\_4 = 1  
 **if** ((found\_idx\_3 == 0) **or** (key\_2[found\_idx\_3-1]==" ")) **and**((found\_idx\_3+len(value[1])==len(key\_2))**or** (key\_2[found\_idx\_3+len(value[1]):found\_idx\_3+len(value[1])+1]**in** (' ',','))):  
 index\_4 = 1   
 found\_idx\_4 = value[1].find(key\_2)  
 **if** found\_idx\_4 ==0 **and** len(value[1])==len(key\_2):  
 index\_4 = 1  
 **if** ((found\_idx\_4 == 0) **or** (value[1][found\_idx\_4-1]==" ")) **and** ((found\_idx\_4 + len(key\_2)== len(value[1])) **or**  
 (value[1][found\_idx\_4+len(key\_2):found\_idx\_4+len(key\_2)+1]**in** (' ',','))):  
 index\_4 = 1   
 **if** index\_3==1 **and** index\_4==1:  
 **break**  
 result\_dic[key].append(index\_3)  
 result\_dic[key].append(index\_4)

此模块编写的是adjust\_results4\_isadog函数，首先打开文件dognames.txt，获取有效的小狗名称，去掉两侧的空格，以小狗名称作为键，1为值，放入dognames\_dict字典中。

与TODO3类似，将result\_dic中的值列表中的索引0（图片的原标签）与dognames字典中的键名作对比，使用两次find函数，将比对结果（0或1）放入列表的索引3中。同理，将索引1（分类器标签）与dognames字典中的键名作对比，使用两次find函数，将比对结果（0或1）放入列表的索引4中。即修改完成results\_dic

TODO5模块

**def** **calculates\_results\_stats**(result\_dic):  
 result\_stats = {}  
 n\_picture = len(result\_dic) #Z图像数量  
 n\_correct\_dogs = 0 #A小狗匹配正确的数量  
 n\_picture\_of\_dogs = 0 #B小狗图像的数量  
 n\_incorrect\_not\_dogs = 0 #C正确非小狗匹配项的数量，两个标签都不是小狗  
 n\_not\_dogs = 0 #D非小狗图像的数量  
 n\_correct\_breed\_and\_dogs = 0 #E正确品种匹配  
 n\_correct\_breed = 0 #Y标签匹配项的数量  
 **for** key,value **in** result\_dic.items():  
 **if** value[3] == 1:  
 n\_picture\_of\_dogs += 1  
 **if** value[3] == 1 **and** value[4] == 1:  
 n\_correct\_dogs += 1  
 **if** value[3] == 0 **and** value[4] == 0:  
 n\_incorrect\_not\_dogs += 1  
 **if** value[3] == 0:  
 n\_not\_dogs += 1  
 **if** value[2] == 1 **and** value[3] == 1:  
 n\_correct\_breed\_and\_dogs += 1  
 **if** value[2] == 1:  
 n\_correct\_breed += 1  
 **if** n\_picture\_of\_dogs == 0 **or** n\_not\_dogs == 0 **or** n\_picture ==0:  
 print("除数为0!!!!!")  
 **return** None  
 **else**:  
 #正确分类的小狗图像所占百分比  
 pct\_correct\_dogs = n\_correct\_dogs / n\_picture\_of\_dogs \* 100  
 #正确分类的非小狗图像所占百分比  
 pct\_correct\_not\_dogs = n\_incorrect\_not\_dogs / n\_not\_dogs \* 100  
 #正确分类的小狗品种所占百分比  
 pct\_correct\_dogs\_sorted = n\_correct\_breed\_and\_dogs / n\_picture\_of\_dogs \* 100  
 #百分比标签匹配数（无论是否为小狗）  
 pct\_correct\_breed = n\_correct\_breed / n\_picture \* 100   
  
 result\_stats['n\_picture'] = n\_picture  
 result\_stats['n\_correct\_dogs'] = n\_correct\_dogs  
 result\_stats['n\_picture\_of\_dogs'] = n\_picture\_of\_dogs  
 result\_stats['n\_incorrect\_not\_dogs'] = n\_incorrect\_not\_dogs  
 result\_stats['n\_not\_dogs'] = n\_not\_dogs  
 result\_stats['n\_correct\_breed\_and\_dogs'] = n\_correct\_breed\_and\_dogs  
 result\_stats['n\_correct\_breed'] = n\_correct\_breed  
 result\_stats['pct\_correct\_dogs'] = pct\_correct\_dogs  
 result\_stats['pct\_correct\_not\_dogs'] = pct\_correct\_not\_dogs  
 result\_stats['pct\_correct\_dogs\_sorted'] = pct\_correct\_dogs\_sorted  
 result\_stats['pct\_correct\_breed'] = pct\_correct\_breed  
 **return** result\_stats

该模块编写calculates\_results\_stats函数，在此函数中，有多个参数：

* + - 1. n\_picture = len(result\_dic) #Z图像数量
      2. n\_correct\_dogs = 0 #A小狗匹配正确的数量
      3. n\_picture\_of\_dogs = 0 #B小狗图像的数量
      4. n\_incorrect\_not\_dogs = 0 #C正确非小狗匹配项的数量，两个标签都不是小狗
      5. n\_not\_dogs = 0 #D非小狗图像的数量
      6. n\_correct\_breed\_and\_dogs = 0 #E正确品种匹配
      7. n\_correct\_breed = 0 #Y标签匹配项的数量

依据每个参数进行相应的计算，将值存入result\_stats字典，键即为变量名：

1. 正确分类的小狗图像所占百分比pct\_correct\_dogs = n\_correct\_dogs / n\_picture\_of\_dogs \* 100
2. 正确分类的非小狗图像所占百分比pct\_correct\_not\_dogs = n\_incorrect\_not\_dogs / n\_not\_dogs \* 100
3. 正确分类的小狗品种所占百分比pct\_correct\_dogs\_sorted = n\_correct\_breed\_and\_dogs / n\_picture\_of\_dogs \* 100
4. 百分比标签匹配数（无论是否为小狗）pct\_correct\_breed = n\_correct\_breed / n\_picture \* 100

TODO6模块

**def** **print\_results**(result\_dic,result\_stats,model\_name,

print\_incorroect\_dogs=False,print\_incorrect\_breed = False):  
 print(f"使用CNN模型架构中的{models[model\_name]}架构，传入的参数是{model\_name}")  
 print(f"图像数量：{result\_stats['n\_picture']}")  
 print(f"小狗图像数量：{result\_stats['n\_picture\_of\_dogs']}")  
 print(f"非小狗图像数量：{result\_stats['n\_not\_dogs']}")  
 print("\n结果:")  
 print(f"\t正确的小狗图像所占百分比：{result\_stats['pct\_correct\_dogs']}%")  
 print(f"\t正确的品种图像所占百分比：{result\_stats['pct\_correct\_dogs\_sorted']}%")  
 print(f"\t正确的非小狗图像所占百分比：{result\_stats['pct\_correct\_not\_dogs']}%")  
 print(f"\t匹配项所占百分比（包含小狗图像和非小狗图像）：{result\_stats['pct\_correct\_breed']}%")  
 **if** print\_incorroect\_dogs:  
 print(f"\n分类错误的项：")  
 print("\n 图片名：")  
 **for** key,value **in** result\_dic.items():  
 **if** print\_incorroect\_dogs:  
 **if** value[2] == 0 :  
 print(f"\t{key.title()}")  
 **if** print\_incorrect\_breed:  
 temp = []  
 print("\n 狗狗品种：")  
 **for** key,value **in** result\_dic.items():  
 **if** value[3] == 1 **and** value[2] == 0 :  
 temp.append(value[0])  
 # 由于狗狗的品种会有重复，使用set集合去除重复项  
 unique = set(temp)  
 **for** dogname **in** unique:  
 print('\t',dogname)

TODO6编写print\_results函数。参数为result\_dic字典，result\_stats（TODO5的返回值），模型名model\_name，以及两个默认参数

该函数主要打印结果。在分类错误项中，考虑到狗狗的品种可能会有重复的，所以使用set集合去除重复项。

实现截图：（如图3-5）

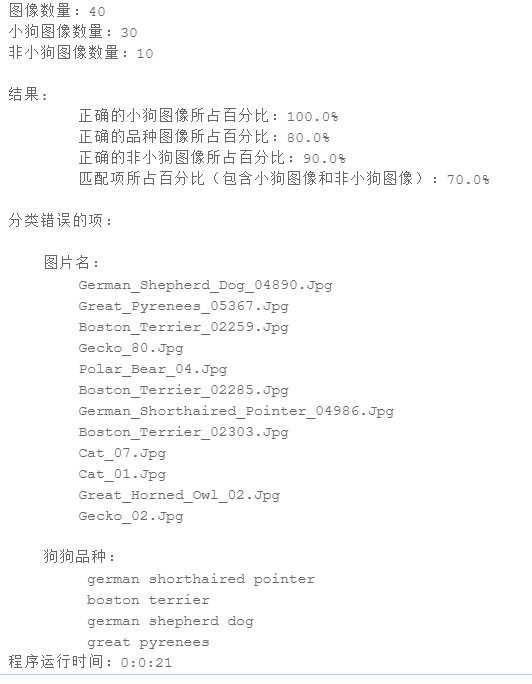


图3- 5

比较模型的效果

**import** seaborn **as** sns  
**import** numpy **as** np  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
   
#构建一个字典total\_result，将模型名称作为键，得到的相关结果作为值  
total\_result = {}  
# 构建一个运行时间的字典，键为CNN模型名称，值为消耗的时间  
total\_time = {}  
**for** key, value **in** models.items():  
 start = time()  
 result\_dic\_1 = classify\_images(answers\_dic,key)  
 adjust\_results4\_isadog(result\_dic\_1,dogfile)  
 total\_result[key] = calculates\_results\_stats(result\_dic\_1)  
 end = time()  
 total = end - start  
 total\_time[key] = total  
  
#一：展示程序运行消耗的时间  
list\_x = []  
list\_y = []  
**for** key, value **in** total\_time.items():  
 list\_x.append(key)  
 list\_y.append(value)  
x = np.array(list\_x)  
y = np.array(list\_y)  
df = pd.DataFrame({"Runnig Time": x,"Time:(s)": y})  
sns.barplot("Runnig Time","Time:(s)",palette="RdBu\_r",data=df)  
plt.savefig("result/running\_time.png")  
plt.show()  
  
#二：正确分类的小狗图像所占百分比对比  
pct\_correct\_dogs\_x\_list = []  
pct\_correct\_dogs\_y\_list = []  
**for** key, value **in** total\_result.items():  
 pct\_correct\_dogs\_x\_list.append(key)  
 pct\_correct\_dogs\_y\_list.append(value['pct\_correct\_dogs'])  
pct\_correct\_dogs\_x = np.array(pct\_correct\_dogs\_x\_list)  
pct\_correct\_dogs\_y = np.array(pct\_correct\_dogs\_y\_list)  
df = pd.DataFrame({"Percentage of correctly classified dogs images":pct\_correct\_dogs\_x,"Percentage:%":pct\_correct\_dogs\_y})  
sns.barplot("Percentage of correctly classified dogs images","Percentage:%",palette="pastel",data=df)  
plt.savefig("result/pecentage\_of\_correctly\_classfied\_dogs\_images.png")  
plt.show()  
  
#三：正确分类的非小狗图像所占百分比对比  
pct\_correct\_not\_dogs\_x\_list = []  
pct\_correct\_not\_dogs\_y\_list = []  
**for** key, value **in** total\_result.items():  
 pct\_correct\_not\_dogs\_x\_list.append(key)  
 pct\_correct\_not\_dogs\_y\_list.append(value['pct\_correct\_not\_dogs'])  
pct\_correct\_not\_dogs\_x = np.array(pct\_correct\_not\_dogs\_x\_list)  
pct\_correct\_not\_dogs\_y = np.array(pct\_correct\_not\_dogs\_y\_list)  
df = pd.DataFrame({"Percentage of correctly classified non-dogs images":pct\_correct\_not\_dogs\_x,"Percentage:%":pct\_correct\_not\_dogs\_y})  
sns.barplot("Percentage of correctly classified non-dogs images","Percentage:%",palette="Set2",data=df)  
plt.savefig("result/pecentage\_of\_correctly\_classfied\_not\_dogs\_images.png")  
plt.show()  
  
#四：正确分类的小狗品种所占百分比对比  
pct\_correct\_dogs\_sorted\_x\_list = []  
pct\_correct\_dogs\_sorted\_y\_list = []  
**for** key, value **in** total\_result.items():  
 pct\_correct\_dogs\_sorted\_x\_list.append(key)  
 pct\_correct\_dogs\_sorted\_y\_list.append(value['pct\_correct\_dogs\_sorted'])  
pct\_correct\_dogs\_sorted\_x = np.array(pct\_correct\_dogs\_sorted\_x\_list)  
pct\_correct\_dogs\_sorted\_y = np.array(pct\_correct\_dogs\_sorted\_y\_list)  
df = pd.DataFrame({'Percentage of correctly classified dogs':pct\_correct\_dogs\_sorted\_x,"Percentage:%":pct\_correct\_dogs\_sorted\_y})  
sns.barplot("Percentage of correctly classified dogs","Percentage:%",palette="Set2",data=df)  
plt.savefig("result/pecentage\_of\_correctly\_classfied\_dogs.png")  
plt.show()  
  
#五：标签匹配数所占百分比对比  
pct\_correct\_breed\_x\_list = []  
pct\_correct\_breed\_y\_list = []  
**for** key, value **in** total\_result.items():  
 pct\_correct\_breed\_x\_list.append(key)  
 pct\_correct\_breed\_y\_list.append(value['pct\_correct\_breed'])  
pct\_correct\_breed\_x = np.array(pct\_correct\_breed\_x\_list)  
pct\_correct\_breed\_y = np.array(pct\_correct\_breed\_y\_list)  
df = pd.DataFrame({"Percentage of the number of tag matches":pct\_correct\_breed\_x,"Percentage:%":pct\_correct\_breed\_y})  
sns.barplot("Percentage of the number of tag matches","Percentage:%",palette="ch:s=.25,rot=-.25",data=df)  
plt.savefig("result/pecentage\_of\_the\_number\_of\_tag\_matches.png")  
plt.show()

在该模块中，调用了seaborn，numpy，pandas和matplotlib库，绘制了以下五幅图：

1. 程序运行消耗的时间
2. 正确分类的小狗图像所占百分比对比
3. 正确分类的非小狗图像所占百分比对比
4. 正确分类的小狗品种所占百分比对比
5. 标签匹配数所占百分比对比
   * 1. **使用Finetune训练模型模块**

由于第二部分和第三部分除了自己找的50张图片不同，其他都是一样的，故合并在此处一起写。

准备工作：

!pip install paddlehub==1.7.0 -i https://pypi.tuna.tsinghua.edu.cn/simple  
# -\*- coding: utf8 -\*-  
**import** paddlehub **as** hub  
%set\_env CPU\_NUM=1  
**import** paddle  
paddle.enable\_static()  
# 选择模型  
# 此处代码为加载Hub提供的图像分类的预训练模型  
module = hub.Module(name="googlenet\_imagenet")

首先导入paddlehub库，然后选择模块一中准确率最高的模型：googlenet\_imagenet

数据准备

# 此处演示的是直接用PaddleHub提供的数据集  
**from** paddlehub.dataset.base\_cv\_dataset **import** BaseCVDataset  
**class** **DemoDataset**(BaseCVDataset):   
 **def** **\_\_init\_\_**(self):   
 # 数据集存放位置  
 self.dataset\_dir = "work/zhongyaocai"  
 super(DemoDataset, self).\_\_init\_\_(  
 base\_path=self.dataset\_dir,  
 train\_list\_file="train\_list.txt",  
 validate\_list\_file="validate\_list.txt",  
 test\_list\_file="test\_list.txt",  
 # predict\_file="predict\_list.txt",  
 label\_list\_file="label\_list.txt",  
 # label\_list=["数据集所有类别"]  
 )  
dataset = DemoDataset()

制作数据集：编写了两个程序，代码如下

程序一：

# -\*- coding: utf-8 -\*-  
**import** os  
labels = []  
**with** open("label\_list.txt") **as** label\_object:  
 label\_list = label\_object.readlines()  
 **for** str\_label **in** label\_list:  
 temp\_list = str\_label.split(' ')  
 labels.append(temp\_list[0])  
  
# 设定文件路径  
**for** label **in** labels:  
 path = 'classification/'+label  
 i = 1  
 # 对目录下的文件进行遍历  
 **for** file **in** os.listdir(path):  
 # 判断是否是文件  
 **if** os.path.isfile(os.path.join(path, file)) == True:  
 # 设置新文件名  
 new\_name=file.replace(file, f"{label}\_used\_for\_classification\_{i}.jpg")  
 # 重命名  
 os.rename(os.path.join(path, file), os.path.join(path, new\_name))  
 i += 1  
# 结束  
print("End")

程序一针对于自己获取的50张图片命名不规范的问题做出了修改，首先程序读取label\_list.txt中的五个标签，分别对应到文件夹中五种中药材。将五种中药材的名字统一改为“药材名”+\_used\_for\_classification\_+“数字”+.jpg的格式。并且移动到对应的文件夹中。

这50张图片命名规范对主程序进行图像分类没有任何影响，只是为了在分类后方便比对是否分类错误。

程序二：

**from** os **import** listdir  
**import** random  
  
**def** **get\_item\_labels**(file):  
 # 返回的是一个列表，获取文件夹里的文件名。  
 filename\_list = listdir(file)  
 itemlabels\_list = []  
 **for** filename **in** filename\_list:  
 itemlabels\_list.append(filename)  
 **return** itemlabels\_list  
  
  
folder\_names = ['baihe','dangshen', 'gouqi',  
 'huaihua', 'jinyinhua']  
  
# 读取所有的图片  
total\_lists = {}  
**for** folder **in** folder\_names:  
 temp\_list = get\_item\_labels(folder)  
 random.shuffle(temp\_list)  
 total\_lists[folder] = temp\_list  
  
label = 0  
**with** open('train\_list.txt', 'w') **as** file\_object\_1:  
 **with** open('test\_list.txt', 'w') **as** file\_object\_2:  
 **with** open('validate\_list.txt', 'w') **as** file\_object\_3:  
 **for** key, value **in** total\_lists.items():  
 num = 0  
 **for** name **in** value:  
 num += 1  
 **if** num <= 120:  
 file\_object\_1.write(key + "/" + name + ' ' + str(label) + '\n')  
 **if** 120 < num <= 150:  
 file\_object\_2.write(key + "/" + name + ' ' + str(label) + '\n')  
 **if** num > 150:  
 file\_object\_3.write(key + "/" + name + ' ' + str(label) + '\n')  
 label += 1

程序二是为了将老师提供的数据集图片标签写入到train\_list.txt、test\_list.txt、validate\_list.txt三个txt文件中。

通过以上两个程序，修改好了自定义数据集以及自己要找的50张中草药图片

中间步骤（包括：生成Reader、选择运行时配置、组建Finetune Task、开始Finetune）

data\_reader = hub.reader.ImageClassificationReader(  
 image\_width=module.get\_expected\_image\_width(), #预期图片经过reader处理后的图像宽度  
 image\_height=module.get\_expected\_image\_height(),#预期图片经过reader处理后的图像高度  
 images\_mean=module.get\_pretrained\_images\_mean(),#进行图片标准化处理时所减均值。默认为None  
 images\_std=module.get\_pretrained\_images\_std(), #进行图片标准化处理时所除标准差。默认为None  
dataset=dataset)  
  
config = hub.RunConfig(  
 use\_cuda=True, #是否使用GPU训练，默认为False；  
 num\_epoch=20, #Fine-tune的轮数；使用4轮，直到训练准确率达到90%多  
 checkpoint\_dir="cv\_finetune\_turtorial\_demo", #模型checkpoint保存路径, 若用户没有指定，程序会自动生成；  
 batch\_size=32, #训练的批大小，如果使用GPU，请根据实际情况调整batch\_size；  
 eval\_interval=50, #模型评估的间隔，默认每100个step评估一次验证集；  
strategy=hub.finetune.strategy.DefaultFinetuneStrategy()) #Fine-tune优化策略；

#获取module的上下文信息包括输入、输出变量以及paddle program  
input\_dict, output\_dict, program = module.context(trainable=True)   
  
#待传入图片格式  
img = input\_dict["image"]   
  
#从预训练模型的输出变量中找到最后一层特征图，提取最后一层的feature\_map  
feature\_map = output\_dict["feature\_map"]   
  
#待传入的变量名字列表  
feed\_list = [img.name]  
  
task = hub.ImageClassifierTask(  
 data\_reader=data\_reader, #提供数据的Reader  
 feed\_list=feed\_list, #待feed变量的名字列表  
 feature=feature\_map, #输入的特征矩阵  
 num\_classes=dataset.num\_labels, #分类任务的类别数量，此处来自于数据集的num\_labels  
config=config) #运行配置  
  
run\_states = task.finetune\_and\_eval() #通过众多finetune API中的finetune\_and\_eval接口，可以一边训练网络，一边打印结果

这部分步骤大部分没有进行修改，保持原样

use\_cuda=True使用GPU训练，num\_epoch=20设置finetune的轮数为20轮。

使用模型进行预测

**import** numpy **as** np  
**import** random  
**import** os  
**import** shutil  
  
# 获取所有图片的路径  
**def** **get\_file\_names**():  
 filenames = os.listdir("work/zhongyaocai/classification/total\_images/")  
 random.shuffle(filenames)  
 new\_filenames = []  
 **for** filename **in** filenames:  
 new\_filename = "work/zhongyaocai/classification/total\_images/"+filename  
 new\_filenames.append(new\_filename)  
 **return** new\_filenames  
  
# 复制文件到指定的文件夹中  
**def** **copyfile**(srcfile, dstfile):  
 **if** **not** os.path.isfile(srcfile):  
 print("%s not exist!" % (srcfile))  
 **else**:  
 fpath, fname = os.path.split(dstfile) # 分离文件名和路径  
 **if** **not** os.path.exists(fpath):  
 os.makedirs(fpath) # 创建路径  
 shutil.copyfile(srcfile, dstfile) # 复制文件  
  
data = get\_file\_names() #此处传入需要识别的照片地址  
# print(data) #测试使用  
label\_map = dataset.label\_dict() # 结果为{0: 'baihe 0', 1: 'dangshen 1', 2: 'gouqi 2', 3: 'huaihua 3', 4: 'jinyinhua 4'}  
index = 0  
label\_folders = {} # 创建一个字典，键为标签（0到4），值为对应的文件夹的名字  
**for** label,value **in** label\_map.items():  
 str\_list = value.split(' ')  
 label\_folders[label] = str\_list[0]  
  
classify\_images\_dir = "work/zhongyaocai/classification/"  
  
# get classification result  
run\_states = task.predict(data=data) #进行预测  
results = [run\_state.run\_results **for** run\_state **in** run\_states] #得到用新模型预测test照片的结果  
  
**for** batch\_result **in** results:  
 # get predict index  
 batch\_result = np.argmax(batch\_result, axis=2)[0]  
 **for** result **in** batch\_result:  
 index += 1  
 # 将图片复制到识别后对应的文件夹中,首先获取文件名，然后创建复制后的路径  
 str\_list = data[index-1].split('/')  
 destFile = classify\_images\_dir + label\_folders[result] + '/' + str\_list[-1]  
 sourceFile = data[index -1]  
 copyfile(sourceFile, destFile)  
 result = label\_map[result]  
 print("input %i is %s, and the predict result is %s" %  
 (index, data[index - 1], result))

首先导入模块numpy、random、os、shutil库

自定义函数：

* get\_file\_names函数：获取所有图片的路径，即自己寻找的50张中药材图片，该文件夹路径为：work/zhongyaocai/classification/total\_images。50张中药材图片都在其中。使用方法os.listdir获取所有的图片名称，再用random.shuffle打乱顺序，体现随机性。将50张随机排序的图片名加上路径，并放在列表new\_filenames中并返回。
* copyfile函数：接受两个参数srcfile和dstfile，srcfile是原来的文件，dstfile是要复制到某处的文件。首先判断文件srcfile是否存在，若存在则将目标文件分离文件名和路径。若路径存在，则将文件srcfile复制到dstfile处。注：该函数是用于复制，而不是移动文件。

首先调用get\_file\_names函数，得到含有50张随机排序的图片名（含路径）的列表data。读取数据集中文件label\_list，返回一个字典label\_map（键值对为：{0: 'baihe 0', 1: 'dangshen 1', 2: 'gouqi 2', 3: 'huaihua 3', 4: 'jinyinhua 4'}）。创建变量index，初始值设置为0，用于确定一共识别了多少张图片。然后创建一个字典：键为标签（0到4），值为对应的文件夹的名字。即：{0: 'baihe', 1: 'dangshen', 2: 'gouqi', 3: 'huaihua', 4: 'jinyinhua'}，创建变量classify\_images\_dir，用于存储用于分类的文件夹的路径

接着进行预测，将data数据传入方法task.predict中，通过解析列表得到结果results。对results进行处理，最后图像识别的结果存储到result（即为数字0~4）中，首先找到源文件路径，即为sourceFile = data[index -1]，然后找到目标路径，即为destFile = classify\_images\_dir + label\_folders[result] + '/' + str\_list[-1]。调用函数copyfile，将源文件复制到目标路径。result = label\_map[result]，将原来的result中数字转化为更清晰易懂的str类型，例：“jinyinhua 4”.并打印出结果

**3.2.3 数据集增强代码展示模块**

创建Img类

**import** math  
**import** matplotlib.pyplot **as** plt  
**import** numpy **as** np  
**from** PIL **import** Image  
  
**def** **DotMatrix**(A, B):

**return** np.matmul(A, B) # 返回矩阵  
  
**class** **Img**:  
 **def** **\_\_init\_\_**(self, image, rows, cols, center=[0, 0]):  
 self.src = image # 原始图像,是一个二维数组，每一个元素都是一个1\*3矩阵  
 self.rows = rows # 原始图像的行  
 self.cols = cols # 原始图像的列  
 self.center = center # 旋转中心，默认是[0,0]  
 self.change\_src\_pos = False # 是否更改旋转中心，用于旋转函数中，其他函数均为False  
 self.angle\_270 = False # 在旋转函数中，判断旋转角度是否为270°  
 self.angle\_90 = False # 在旋转函数中，判断旋转角度是否为90°  
  
 **def** **Move**(self, delta\_x, delta\_y):  
 '''  
 本函数处理生成做图像平移的矩阵  
 '''  
 self.transform = np.array([[1, 0, delta\_x],  
 [0, 1, delta\_y],  
 [0, 0, 1]])  
 self.change\_src\_pos = False  
  
 **def** **Zoom**(self, factor): # 缩放  
 # factor>1表示缩小；factor<1表示放大  
 self.transform = np.array([[factor, 0, 0],  
 [0, factor, 0],  
 [0, 0, 1]])  
 self.change\_src\_pos = False  
  
 **def** **Horizontal**(self):

# 水平镜像  
 self.transform = np.array([[-1, 0, self.rows],  
 [0, 1, 0],  
 [0, 0, 1]])  
 self.change\_src\_pos = False  
  
 **def** **Vertically**(self):  
 # 垂直镜像  
 self.transform = np.array([[1, 0, 0],  
 [0, -1, self.cols],  
 [0, 0, 1]])  
 self.change\_src\_pos = False  
  
 **def** **Rotate**(self, beta): # 旋转，传入的值是数字，不是弧度  
 # beta>0表示逆时针旋转；beta<0表示顺时针旋转  
 **if** (beta + 90) % 360 == 0:  
 self.angle\_270 = True  
 **elif** (beta - 90) % 360 == 0:  
 self.angle\_90 = True  
 **else**:  
 self.angle\_270 = False  
 self.angle\_90 = False  
 beta = math.radians(beta) #将数字转化为弧度  
 self.transform = np.array([[math.cos(beta), math.sin(beta), 0],  
 [-math.sin(beta), math.cos(beta), 0],  
 [0, 0, 1]])  
 self.change\_src\_pos = True # 改变旋转的中心  
  
 **def** **Process**(self):  
 **if** self.change\_src\_pos: # 用于旋转  
 # 初始化定义目标图像，具有3通道RBG值  
 **if** self.angle\_270 **is** False **and** self.angle\_90 **is** False:  
 self.dst = np.zeros((self.rows, self.cols, 3), dtype=np.uint8)  
 **else**: #如果是旋转270°或90°，则需要修改图片的大小，将图片的像素高和宽互换  
 self.dst = np.zeros((self.cols, self.rows, 3), dtype=np.uint8)  
 # 提供for循环，遍历图像中的每个像素点，然后使用矩阵乘法，找到变换后的坐标位置  
 **for** i **in** range(self.rows):  
 **for** j **in** range(self.cols):  
 src\_pos = np.array([i - self.rows / 2, j - self.cols / 2, 1]) # 设置原始坐标点矩阵  
 [x, y, z] = DotMatrix(self.transform, src\_pos) # 和对应变换做矩阵乘法  
 x = int(x) + int(self.rows / 2)  
 y = int(y) + int(self.cols / 2)  
  
 **if** self.angle\_270 **is** False **and** self.angle\_90 **is** False:  
 # 旋转角度不是270°或90°，采用一般的旋转方法，生成的图片会有部分空白（180°和360°除外）  
 **if** x >= self.rows **or** y >= self.cols **or** x < 0 **or** y < 0:  
 self.dst[i][j] = 255 # 处理未落在原图像中的点的情况  
 **else**:  
 self.dst[i][j] = self.src[x][y] # 使用变换后的位置  
 **elif** self.angle\_90:  
 # 旋转90°，生成的图片不会存在空白  
 self.dst[j][i] = self.src[i][self.cols - 1 - j]  
 **elif** self.angle\_270:  
 # 旋转270，生成的图片不会存在空白  
 self.dst[j][i] = self.src[i][j]  
 **else**: # 其他的操作：镜像，放缩，平移  
 # 初始化定义目标图像，具有3通道RBG值  
 self.dst = np.zeros((self.rows, self.cols, 3), dtype=np.uint8)  
 # 提供for循环，遍历图像中的每个像素点，然后使用矩阵乘法，找到变换后的坐标位置  
 **for** i **in** range(self.rows):  
 **for** j **in** range(self.cols):  
 src\_pos = np.array([i - self.center[0], j - self.center[1], 1]) # 设置原始坐标点矩阵  
 [x, y, z] = DotMatrix(self.transform, src\_pos) # 和对应变换做矩阵乘法  
 x = int(x) + self.center[0]  
 y = int(y) + self.center[1]  
 **if** x >= self.rows **or** y >= self.cols **or** x < 0 **or** y < 0:  
 self.dst[i][j] = 255 # 处理未落在原图像中的点的情况  
 **else**:  
 self.dst[i][j] = self.src[x][y] # 使用变换后的位置

首先导入模块math、matplotlib、numpy、PIL中的Image，接着定义函数DotMatrix，用于矩阵的乘法，返回的是结果。创建类Img，属性有

1. self.src：用于存储原始图像，是一个二维的数组，每一个元素都是一个1\*3的矩阵
2. self.rows：用于存储原始图像的行
3. self.cols：用于存储原始图像的列
4. self.center：旋转中心的坐标，默认是[0,0]
5. self.change\_src\_pos：是否更改旋转中心，默认为False
6. self.angle\_270：在旋转函数中，判断旋转角度是否为270°
7. self.angle\_90：在旋转函数中，判断旋转角度是否为90°

定义的方法有：

1. Move：用于处理图像平移
2. Zoom：用于处理图像缩放
3. Horizontal：用于处理图像水平镜像
4. Vertically：用于处理图像垂直镜像
5. Rotate：用于处理图像旋转
6. Process：用于图像的处理

测试程序：

**if** \_\_name\_\_ == '\_\_main\_\_':  
 infer\_path = r'sample\_picture.png' # 示例图片  
 imgv = Image.open(infer\_path) # 打开图片  
 plt.imshow(imgv) # 根据数组绘制图像  
 print("原图像")  
 plt.show() # 显示图像  
 rows = imgv.size[1]  
 cols = imgv.size[0]  
 print(rows, cols) # 注意此处rows和cols的取值方式，获得的是图片的像素高和宽  
 imgv = np.array(imgv) # 从图像生成数组  
 img = Img(imgv, rows, cols, [0, 0]) # 生成一个自定Img类对象[0,0]代表处理的中心点  
 #以下操作为图像的绘制  
 # 1.旋转10°  
 img.Rotate(10) # 旋转10°  
 img.Process() # 进行矩阵变换  
 img1 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img1)  
 plt.savefig("pictures/rotate\_10\_angle.png") # 保存图片  
 print("逆时针旋转图片10°")  
 plt.show()  
  
 # 2.旋转90°  
 img.Rotate(90) # 旋转90°  
 img.Process() # 进行矩阵变换  
 img2 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img2)  
 plt.savefig("pictures/rotate\_90\_angle.png") # 保存图片  
 print("逆时针旋转图片90°")  
 plt.show()  
  
 # 3.旋转270°  
 img.Rotate(270) # 旋转270°  
 img.Process() # 进行矩阵变换  
 img3 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img3)  
 plt.savefig("pictures/rotate\_270\_angle.png") # 保存图片  
 print("逆时针旋转图片270°")  
 plt.show()  
  
 # 4.旋转180°  
 img.Rotate(180) # 旋转180°  
 img.Process() # 进行矩阵变换  
 img4 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img4)  
 plt.savefig("pictures/rotate\_180\_angle.png") # 保存图片  
 print("逆时针旋转图片180°")  
 plt.show()  
  
 # 5.镜像  
 img.Vertically() #垂直镜像(0,0)  
 img.Process() # 进行矩阵变换  
 img5 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img5)  
 plt.savefig("pictures/vertical.png") # 保存图片  
 print("垂直镜像")  
 plt.show()  
  
 # 6.镜像  
 img.Horizontal() #水平镜像(0，0)  
 img.Process() # 进行矩阵变换  
 img6 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img6)  
 plt.savefig("pictures/horizontal.png") # 保存图片  
 print("水平镜像")  
 plt.show()  
  
 # 7.平移  
 img.Move(-500,-500) #平移(-500,-500)  
 img.Process() # 进行矩阵变换  
 img7 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img7)  
 plt.savefig("pictures/move.png") # 保存图片  
 print("平移图片")  
 plt.show()  
  
 # 8.平移  
 img.Zoom(0.5) #缩放  
 img.Process() # 进行矩阵变换  
 img8 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img8)  
 plt.savefig("pictures/zoom.png") # 保存图片  
 print("放大图片")  
 plt.show()

* 1. 此处的测试程序是用于测试每种函数是否封装完整，能否正常运行
  2. 首先传入示例图片路径，通过Image处理，再用matplotlib库中函数imshow和show显示原图像
  3. 通过处理imgv生成数组imgv。创建Img类的实例img
  4. 图像操作：
     1. 旋转10°
     2. 旋转90°
     3. 旋转270°
     4. 旋转180°
     5. 垂直镜像
     6. 水平镜像
     7. 平移
     8. 缩放

主程序

**def** **rotate\_10\_Angle**(img, fileFolderName, i):  
 # 1.旋转10°  
 img.Rotate(10) # 旋转10°  
 img.Process() # 进行矩阵变换  
 img1 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img1)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **rotate\_90\_Angle**(img, fileFolderName, i):  
 # 2.旋转90°  
 img.Rotate(90) # 旋转90°  
 img.Process() # 进行矩阵变换  
 img2 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img2)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **rotate\_270\_Angle**(img, fileFolderName, i):  
 # 3.旋转270°  
 img.Rotate(270) # 旋转270°  
 img.Process() # 进行矩阵变换  
 img3 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img3)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **rotate\_180\_Angle**(img, fileFolderName, i):  
 # 4.旋转180°  
 img.Rotate(180) # 旋转180°  
 img.Process() # 进行矩阵变换  
 img4 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img4)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **vertical**(img, fileFolderName, i):  
 # 5.垂直镜像  
 img.Vertically() # 垂直镜像(0,0)  
 img.Process() # 进行矩阵变换  
 img5 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img5)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **horizontal**(img, fileFolderName, i):  
 # 6.水平镜像  
 img.Horizontal() # 水平镜像(0，0)  
 img.Process() # 进行矩阵变换  
 img6 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img6)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **move**(img, fileFolderName, i):  
 # 7.平移  
 img.Move(-30, -30) # 平移(-500,-500)  
 img.Process() # 进行矩阵变换  
 img7 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img7)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **zoom**(img, fileFolderName, i):  
 # 8.缩放  
 img.Zoom(0.5) # 缩放  
 img.Process() # 进行矩阵变换  
 img8 = Image.fromarray(img.dst) # 从处理后的数组生成图像  
 plt.imshow(img8)  
 plt.axis('off')  
 plt.savefig("dataset/" + fileFolderName + "/" + fileFolderName + "\_" + str(i) + ".jpg", bbox\_inches='tight',  
 pad\_inches=0) # 保存图片  
  
**def** **folderName\_fileNumbers**(folderName, picture\_number\_dict):  
 picture\_number\_dict[folderName] = picture\_number\_dict[folderName] + 1  
 **return** picture\_number\_dict[folderName]  
  
**def** **draw\_one\_pitures**(img, folderName, picture\_number\_dict):  
 i = folderName\_fileNumbers(folderName, picture\_number\_dict)  
 rotate\_180\_Angle(img, folderName, i)  
 i = folderName\_fileNumbers(folderName, picture\_number\_dict)  
 vertical(img, folderName, i)  
 i = folderName\_fileNumbers(folderName, picture\_number\_dict)  
 horizontal(img, folderName, i)  
 i = folderName\_fileNumbers(folderName, picture\_number\_dict)  
 zoom(img, folderName, i)  
 i = folderName\_fileNumbers(folderName, picture\_number\_dict)  
 move(img, folderName, i)  
**if** \_\_name\_\_ == '\_\_main\_\_':  
 pictures\_list = []  
 picture\_path\_dict = {}  
 picture\_number\_dict = {}  
 categories = ['dangshen', 'baihe', 'jinyinhua', 'gouqi', 'huaihua']  
 **for** category **in** categories:  
 pictures\_list.extend(listdir("dataset/" + category))  
 picture\_number\_dict[category] = len(listdir("dataset/" + category))  
 **for** picture **in** pictures\_list:  
 picture\_path\_dict[picture] = 'dataset/' + picture.split('\_')[0] + "/" + picture  
 **for** key, value **in** picture\_path\_dict.items():  
 infer\_path = value # 示例图片  
 imgv = Image.open(infer\_path) # 打开图片  
 plt.imshow(imgv) # 根据数组绘制图像  
 rows = imgv.size[1]  
 cols = imgv.size[0]  
 imgv = np.array(imgv) # 从图像生成数组  
 img = Img(imgv, rows, cols, [0, 0]) # 生成一个自定Img类对象[0,0]代表处理的中心点  
 **try**:  
 draw\_one\_pitures(img, key.split('\_')[0], picture\_number\_dict)  
 print(f"图像{key}的增强数据集已绘制成功！")  
 **except**:  
 print(f"图像{key}的增强数据集绘制失败！")

在运行主程序之前，为了主程序的简洁，故将图像处理的每种操作都封装成一个独立的函数，在函数draw\_one\_pitures中，将图像处理的每个函数都装入该函数中，在主程序中只需运行该函数即可完成对一张照片的5种操作。

为了对原数据集中的每张照片都做图像处理。首先调用os库中的listdir函数，获得五个中药材图片文件夹下所有图片，使用方法extend将得到的列表整合在一个共同的列表pictures\_list中。由于获取到的列表都是文件的名称，而不是文件的路径，为此，遍历pictures\_list中的每一个元素，为每个文件都加上路径，作为值，原文件名作为键，存储在字典picture\_path\_dict。定义一个字典picture\_number\_dict，以文件夹名为键，文件夹内的文件数量为值，存储5个键值对，用于后续生成新图片时为新图片命名（假设baihe文件有180个文件，新生成的图片就从181开始命名）。

遍历字典picture\_path\_dict的键值对，对其中的每个键值对，值就是文件的路劲，利用文件路径绘制出5中图片。考虑到每个文件的命名问题，故定义一个函数folderName\_fileNumbers，传入的参数时文件夹名称（folderName），以及以文件夹名为键，文件夹内的文件数量为值的字典picture\_number\_dict，每生成一张新图片，使用方法split将文件名（baihe\_1.jpg）中baihe字符串提取出来，判断为是baihe文件夹下的文件，故将picture\_number\_dict中baihe键对应的值加一，并以新的值作为后缀命名（baihe\_181.jpg）。

在主程序中，每成功绘制一次原数据集中的图片，都会打印图像xxx的增强数据集已绘制成功。若绘制失败，则打印图像xxx的增强数据集绘制失败。

遍历所有文件后，新的增强数据集即绘制成功。

最后，再将新生成的数据集进行处理，修改数据集中的txt文件，上传到第三部分中运行。

1. **复杂工程问题分析**

**4.1第一部分预训练模型问题及解决方案**

1. 模型使用
   1. 问题描述：调用模型时不知道怎么使用模型
   2. 解决方法：查阅官方文档资料（网址为：https://aistudio.baidu.com/aistudio/projectdetail/361892），获得了示例代码：

**import** paddlehub **as** hub  
module = hub.Module(name="alexnet\_imagenet")  
test\_img\_path = "/PATH/TO/IMAGE/"  
# set input dict  
input\_dict = {"image": [test\_img\_path]}  
# execute predict and print the result  
results = module.classification(data=input\_dict)  
**for** result **in** results:  
 print(result)

1. resnet模型
   1. 问题描述：resnet模型准确率为0，图片识别的结果与图片的标签完全匹配不上
   2. 解决方法：查找其他的模型，最后使用新的模型：googlenet\_imagenet和mobilenet\_v2\_imagenet
2. find函数
   1. 问题描述：考虑到指导PDF中的字符串不匹配的情况有两种，一种原标签为cat，分类器标签为skunk, polecat, wood pussy and lynx catamount，并不匹配。还有一种是原标签为german shepherd dog，分类器标签为german shepherd，应该匹配，但是用指导PDF中的方法并不匹配
   2. 解决方法：不仅在原标签中使用find函数寻找分类器标签，还在分类器标签中使用find函数寻找原标签，满足任意一种即可。
3. 输出结果
   1. 问题描述：在主程序输出错误的项时，输出错误的狗狗品种，对于某些模型，可能某一类的狗狗都会识别错误，此时输出的结果就会有重复的项
   2. 解决方法：为了消除重复的项，考虑到集合的唯一性，对识别错误狗狗的品种的列表temp使用set集合去除重复项：

# 由于狗狗的品种会有重复，使用set集合去除重复项  
unique = set(temp)  
**for** dogname **in** unique:  
 print('\t',dogname)

**4.2第二部分使用Finetune问题及解决方案**

1. 数据准备
   1. 问题描述：在baihe文件夹中的文件并没有200张，但是图片的编号达到了200多
   2. 解决方案：重新命名baihe文件夹中的所有文件。使用for循环。
2. predict\_file
   1. 问题描述：在数据准备中，添加指导word中的代码predict\_file="predict\_list.txt"时，程序报错。
   2. 解决方案：询问老师，应该是继承的类中删去了这个属性。将这一行代码注释掉，不使用predict\_list.txt
3. 将识别的文件移动到指定处
   1. 问题描述：不知道怎么将文件移动
   2. 解决方案：在网上查找资料，自己定义了一个函数copyfile()，将文件复制到指定的路径
4. 文件数量
   1. 问题描述：文件的总数超过了1000，即超过了上传的上限
   2. 解决方案：与老师沟通后，先将程序运行，然后将数据集压缩，与其他的文件一起上传。

**4.3数据集增强板块问题及解决方案**

1. 绘制图像
   1. 问题描述：绘制图像时有时会无法显示图像，或多张图像显示部分
   2. 解决方案：切换网络，不使用学校的uestc-wifi连接，使用手机流量开热点连接
2. 旋转问题
   1. 问题描述：在旋转的过程中，当旋转到90°的倍数时，考虑到图像是正的，周围没有空白（不像旋转10°时图像是斜的，图像旁边有部分空白），故想要修改原有代码，考虑到特殊的情况
   2. 解决方案：将原来传递的参数（弧度）更改为新的参数（数字。例：90代表90°）。使用if-elif-else判断角度是否为90的倍数，是90的多少倍。在不同的情况下修改图片大小的格式
3. **团队协作和人员实现展示**

**5.1 团队协作方式**

1. 我们小组采取课堂上相互讨论指出项目bug，课堂下整合资料的方式协作。在课堂上，我们负责写自己的代码，出现问题时会相互交流讨论，在网上查找解决方案，如果网上没有，就会去请教老师。在课堂外，我们会聚在一起，分享自己代码中的易错点。小组分工在下面人员完成情况展示中。
2. 心得：小组合作减轻了每一个人的负担，让我们可以专注于负责自己的模块，而不用去关心多余的事情。此外，小组分工中，与小组成员的合作也使我更进一步意识到了团队合作的重要性，也让我们明白了在与小组成员共同编写程序时，要注意格式的规范，增强程序的可读性。
3. 小组得分：（按比例：陈霆润：2，赵之航：1，幸锴文：1）

**5.2 人员完成情况展示**

TODO5+TODO6的运行结果内容：（因为TODO6就是打印TODO5的内容，故以下的截图只有TODO6）

陈霆润：（（如图5-2-1（a）-（e)及图5-2-2）

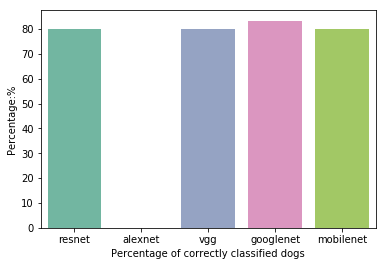
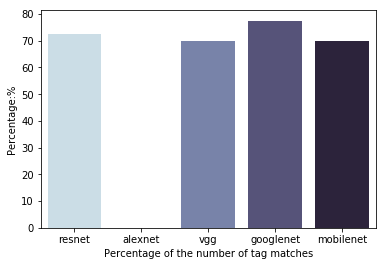


图5-2-1(a) 图5-2-1(b)

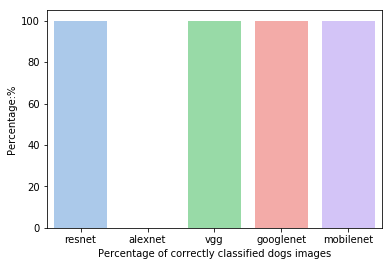
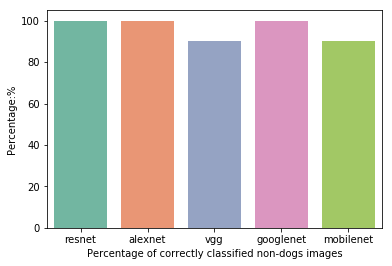


图5-2-1（c） 图5-2-1（d）

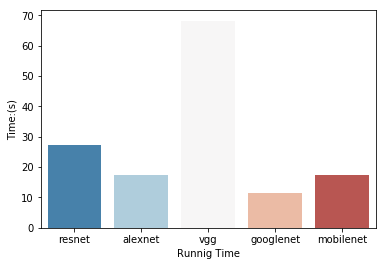


图5-2-1（e）



图5-2-2

赵之航：（如图5-2-3）

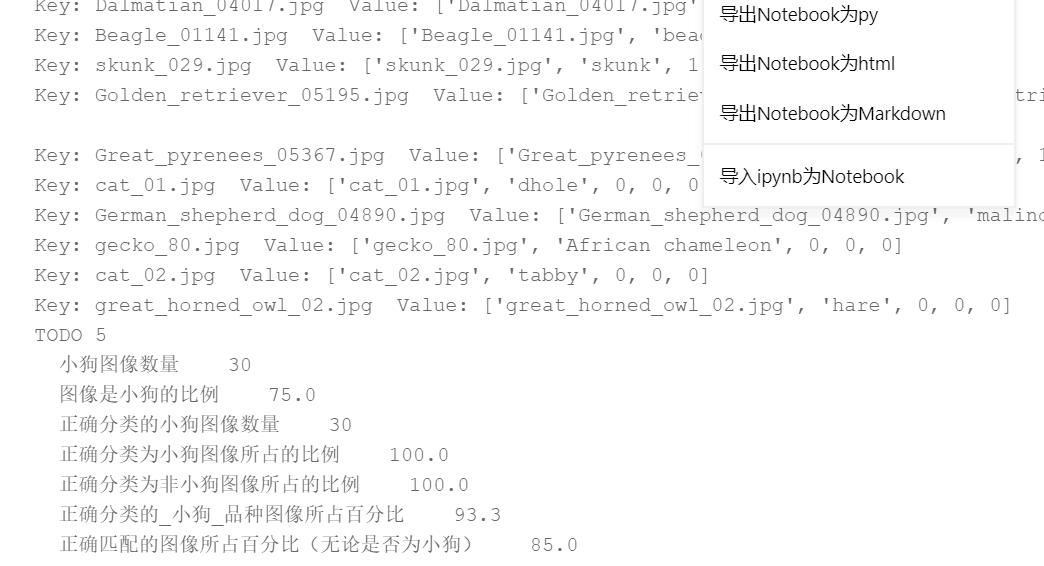


图5-2-3

幸锴文：（如图5-2-4）



图5-2-4

小组分工：（针对于任务使用Finetune和任务数据集增强）

陈霆润：

1. 编写两个小程序，一是为了修改baihe文件夹中的文件命名不规范，二是为了将50张图片重命名，便于主程序运行后对照结果。
2. 负责数据集增强中的类的旋转操作和处理操作方法
3. 主函数中的旋转10、90、270、180四种图像的绘制
4. 负责最终报告

赵之航：

1. 查找2组不同的50张中药材图片，以及数据集的相关制作
2. 负责数据集增强的矩阵乘法、类的平移操作和缩放操作方法
3. 主函数中对原图像的处理、绘制，以及平移图像绘制、缩放图像绘制

幸锴文：

1. 编写函数get\_file\_names，获取50张图片的路径，并存储在data列表中；
2. 编写函数copyfile，将文件复制到指定处
3. 负责数据集增强中的类的水平镜像和垂直镜像方法
4. 主函数中的水平镜像图像绘制、垂直镜像图像绘制

任务二的准确率：（如图5-2-5(a)）

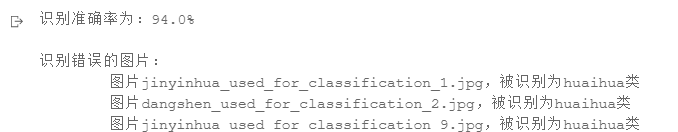


图5-2-5(a)

条形统计图如下：（如图5-2-5(b)）

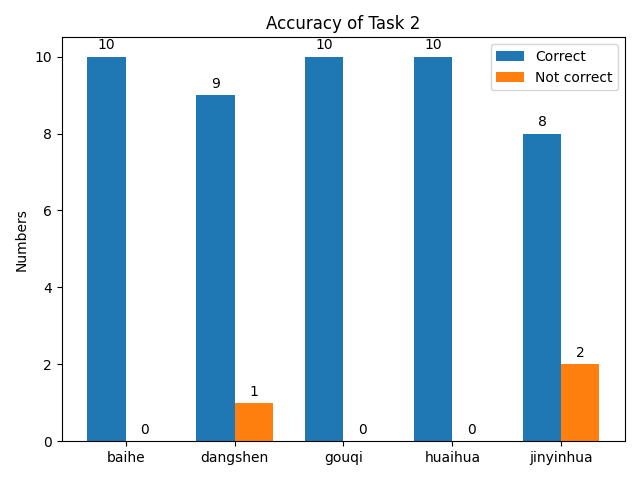


图5-2-5(b)

任务三准确率：（如图5-2-6(a)）

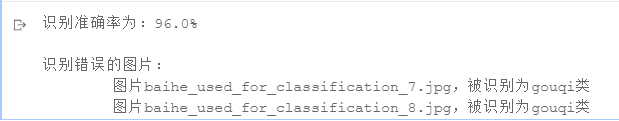


图5-2-6(a)

条形统计图如下：（如图5-2-6(b)）

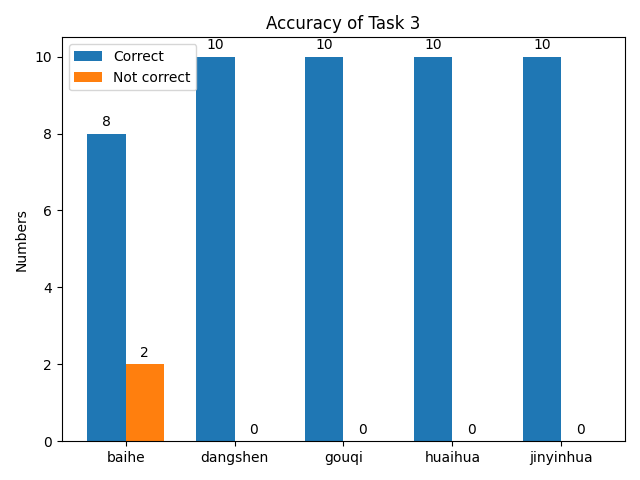


图5-2-6(b)

任务二实现效果截图：（如图5-2-7(a) 和图5-2-7(b)））

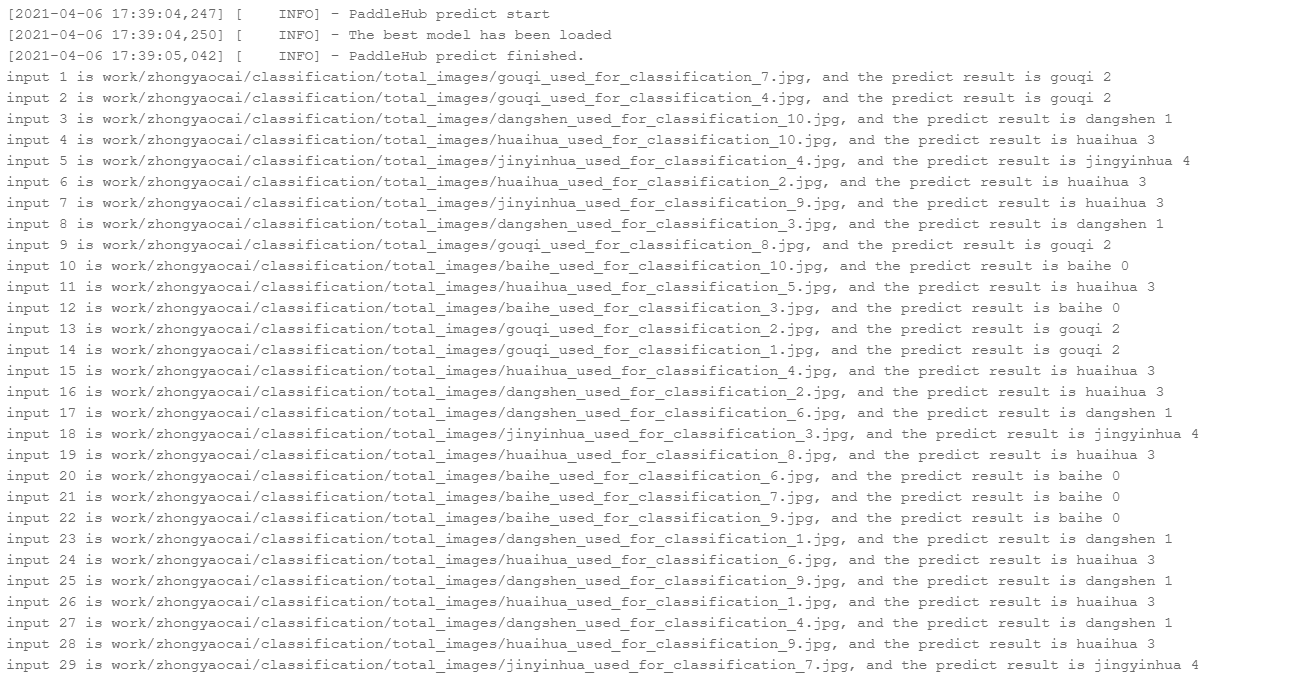


图5-2-7(a)

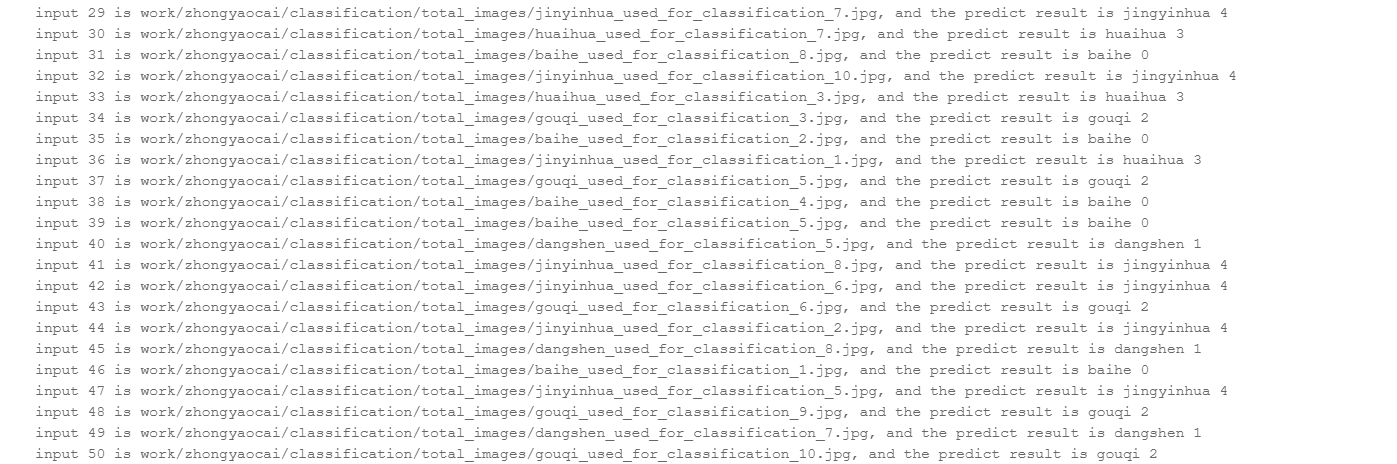


图5-2-7(b)

任务四实现效果截图：（如图5-2-8(a)- (i)）

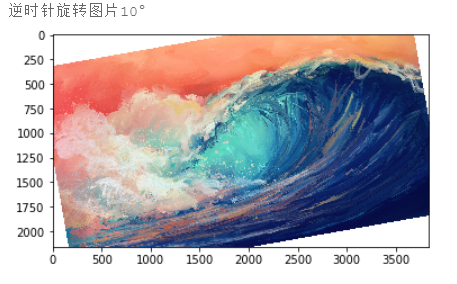
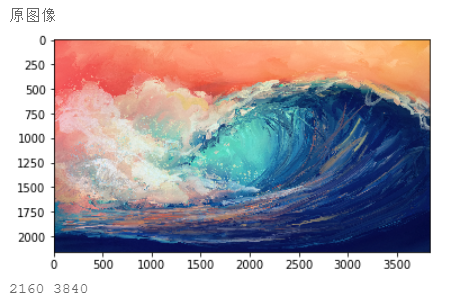


图5-2-8(a) 原图像 图5-2-8(b) 逆时针旋转10°

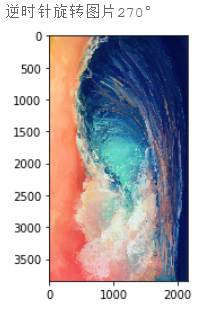
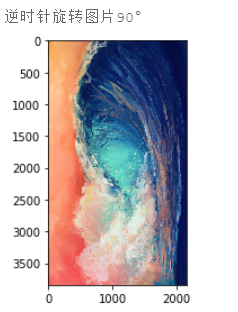


图5-2-8(c) 逆时针旋转90° 图5-2-8(d)逆时针旋转270°

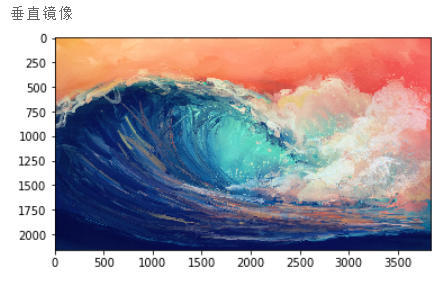
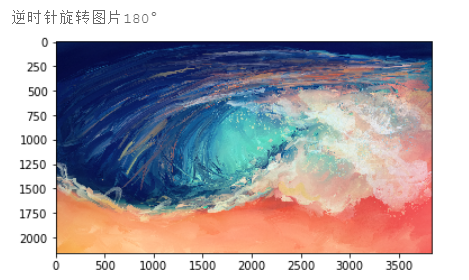


图5-2-8(e) 逆时针旋转180° 图5-2-8(f) 垂直镜像

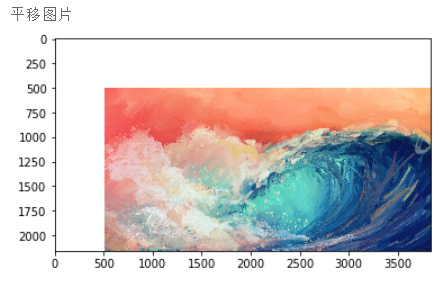
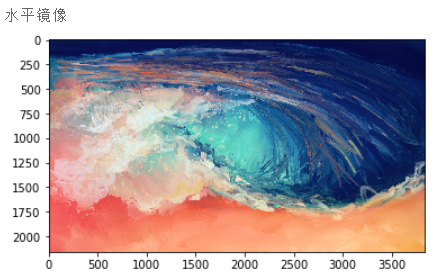


图5-2-8(g) 水平镜像 图5-2-8(h) 平移图片

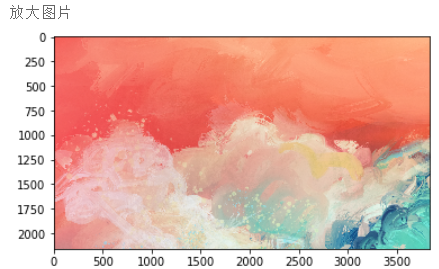


图5-2-8(i) 放大图片

1. **总结**

参加这门课程，对于刚开始学习Python的我来说，具有很大的挑战。在刚开始的时候面对老师讲的人工智能的相关知识感到十分迷茫，更不用说刚开始上机写项目时的手足无措。从零到一的突破往往是最难的，不会使用jupyter notebook，不明白各个板块之间的联系，往下一翻，看到的代码令自己头大。第一个任务TODO1的代码就让我感到迷茫，不知道从哪里开始，要调用哪些函数，对此一无所知。没有办法，只好看着指导PDF来写，照着PDF一步一步地完成自己的代码。然而，当写完TODO1时，心里有一种说不出的喜悦，就像一道许久没有做出来的数学题突然被解出来，大脑突然顿悟的豁然开朗。然后后面就沉浸到这种状态中，一口气做到TODO3。面对从未使用过的模型的调用，我开始阅读示例代码，理清每一步的步骤，又一次的感受到了这种喜悦。在这之后，在实现零的突破后，后面就很快速了。

对于这个课程中的几项作业中的代码，在做的过程中有或多或少的困惑的地方，遇到不懂，会上网查，自己动手解决问题，若是网上也没有，也会找老师请教。就是在这样的磕磕绊绊的过程中，我一步一步完成了曾经让我毫无头绪的代码。不仅学会了调用多种第三方库和标准库，还学会了如何在网上快速找到问题的解答

我很喜欢这门课程，不同于我在网上看的网课教学全部都讲，也不同于以前学习的囿于定式的课程，这门课程将我现阶段理解不了的代码写了出来，而那些相对容易的代码空着让我自己补充，旁边又有markdown解释该步骤，是一种循循诱导的过程，一步一步地带我完成这块拼图。每当完成一个子程序并运行成功时，都是一种难以名状的喜悦。此外，课程中的非唯一答案也让我有更大发挥空间。我可以在第一部分中调用seaborn库画指定的条形图。不像icoding作业中的固定步骤和固定答案，这门课程中每个项目都让我的思维发散，不拘泥于形式，让我思考如何将我的输出美化（修改输出格式），让我思考怎样节省代码，甚至当我认为原来的代码不够好时（有些情况没有考虑到），我可以修改原来的代码，改成我想要实现的代码。这门不拘泥于形式的课程真正的感受到了做项目的感觉，没有固定的答案，没有最好的实现，只有更好的实现，一步一步将自己的实现做的更加完美。

最后，我十分感谢杨老师。谢谢杨老师不厌其烦地为我解答各种各样的问题，为我指出一些细碎的错误。当然，也少不了我的组员的帮助，他们为我指出了我编写程序中的错误，谢谢他们的包容。